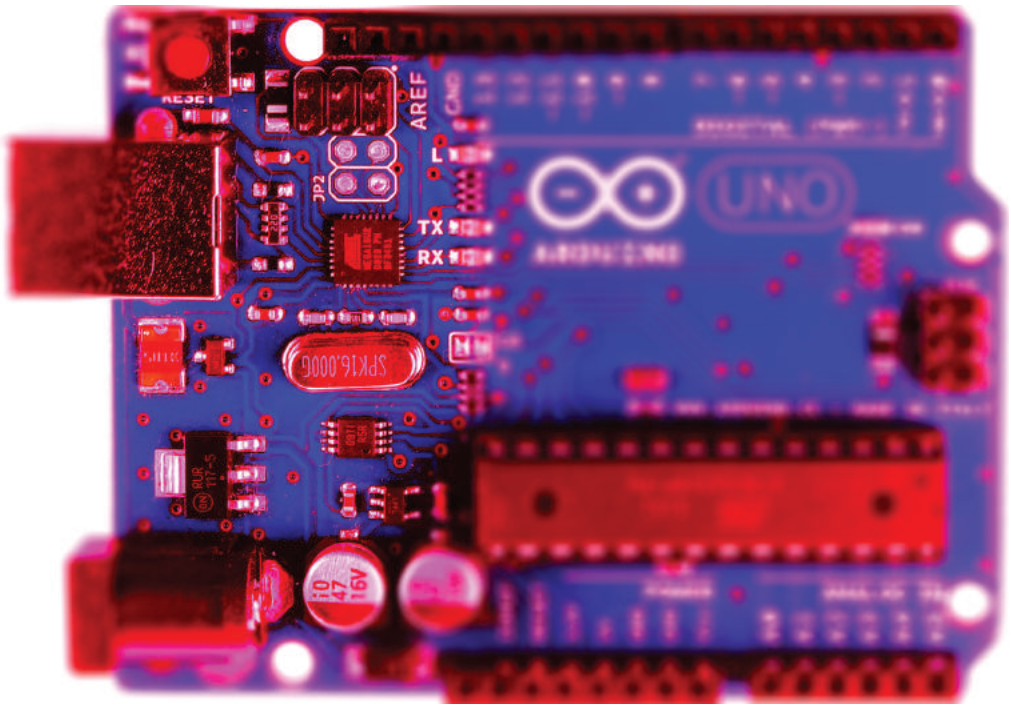
Arduino uno para prototipado rápido: Generador de números aleatorios

G.M. Reyna-Obregón, J.L. Soriano-Hernández, M.E. Leal Miranda,
J.C. Herrera-Lozada, H. Taud, J. A. Álvarez-Cedillo *



*Instituto Politécnico Nacional. Centro de Innovación y Desarrollo Tecnológico en Cómputo.
Maestría en Tecnología de Cómputo, Laboratorio de Tecnología de Computación Inteligente.
Unidad profesional Adolfo López Mateos, México, D.F.
gmarcelro@hotmail.com, jen_lid_hdz@hotmail.com, locus1232000@hotmail.com;
jlozada@ipn.mx, htaud@ipn.mx, jaalvarez@ipn.mx

Recibido: 30 de enero de 2013.
Aceptado: 08 de mayo de 2013.

Resumen
El presente artículo muestra el diseño de un generador de números pseudoaleatorios en un intervalo de 0 hasta 255, estos números tienen la característica de que son generados por ruido del exterior, con lo que se garantiza que la semilla inicial sea aleatoria. El sistema es de propósito general por lo que puede adaptarse sin cambios drásticos a cualquier aplicación que requiera de muestras aleatorias para su funcionamiento y se elaboró usando la tarjeta de desarrollo Arduino Uno, que es un sistema embebido con base en un microcontrolador. Este generador de números aleatorios surge como parte de un proyecto que concierne heurísticas evolutivas y su implementación en hardware.

Palabras clave: números pseudoaleatorios, números aleatorios, ruido, semilla, Arduino Uno.

Abstract
This paper presents the design of a pseudorandom number generator in a range from 0 to 255, these numbers have the feature that they are generated by outside noise, so we guarantee that the initial seed is a random number. The system is general purpose so that it can adapt without dramatic changes to any application requiring random samples for operation and it was developed using the Arduino Uno development board, which is an embedded system based on a microcontroller. This random number generator comes as part of a project concerning evolutionary heuristics and its implementation in hardware.

Key words: pseudorandom number, random number, noise, seed, Arduino Uno.

Introducción

Se dice que una muestra tiene la propiedad de aleatoriedad si todo elemento del conjunto universal o población tiene las mismas posibilidades de ser seleccionado como parte de dicha muestra [1]. Existen diversos métodos para generar números aleatorios. Un generador de verdaderos números aleatorios (RRNG) en hardware utiliza características no predecibles de elementos físicos (radiación de un núcleo atómico, inestabilidad de un oscilador libre, ruido de una resistencia eléctrica, resonancia de las partículas de un rayo láser, entre otros). Los generadores de números pseudoaleatorios (PRNG) son más comunes en hardware debido principalmente, en la mayoría de los casos, a su facilidad de implementación. Los números pseudoaleatorios se generan a partir de una función determinista pero aparentan ser aleatorios. Estos generadores tienen el inconveniente de trabajar con sucesiones periódicas y que a partir de un mismo valor inicial (semilla) se genera siempre la misma sucesión [2].

En general cuando se requiere una impredecibilidad en determinados datos, para alguna aplicación en particular que así lo requiera, se utilizan números aleatorios, debido a que éstos permiten a los modelos matemáticos representar un comportamiento más apegado a la realidad. Para lograr una mejor aproximación a este comportamiento, nuestra herramienta predictiva debe funcionar de manera impredecible, sin conocimiento a priori o tendencia alguna. De esta necesidad surgieron modelos de simulación complejos que inciden en la vida cotidiana como por ejemplo en los juegos de azar, en el comportamiento de la caída de los copos de nieve en una animación por computadora, en un test para localización de errores en circuitos integrados, en la transmisión de datos desde un satélite, en las finanzas, en los sistemas criptográficos y en varios otros que dependen de situaciones no determinísticas.

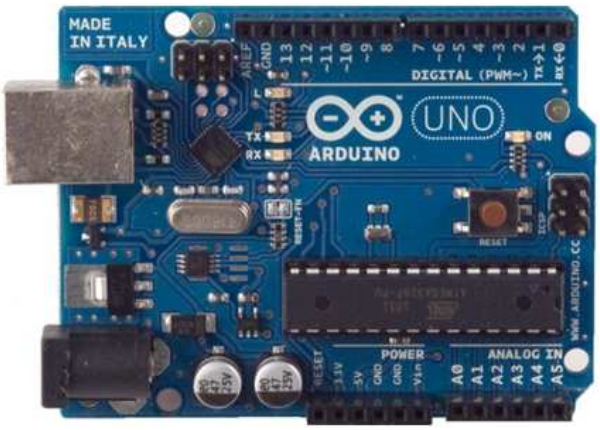


Figura 1: Placa Arduino Uno.

Números pseudoaleatorios

Los números pseudoaleatorios son generados por medio de una función (determinista, no aleatoria) y que aparentan solamente ser aleatorios. Estos números se producen a partir de un valor inicial (semilla) aplicando iterativamente la función generadora que es matemáticamente determinística; esta es la razón por la cual, teniendo el valor de la semilla y la función generadora, es posible conocer o pre-

decir la secuencia de números perdiendo la naturaleza aleatoria. Computacionalmente, la mayoría de los lenguajes de programación incluyen la función *random ( )*, cuyo propósito es generar números aleatorios, aunque a decir verdad, se trata de pseudoaleatorios. Para mejorar el propósito de esta función en un sistema de cómputo convencional, la semilla se genera utilizando el reloj del sistema, para que por lo menos el inicio de la secuencia sea aleatorio. Cabe menciona que los números pseudoaleatorios pueden tener cierto grado de calidad con respecto a la variación en sus muestras, por lo mismo deben ser sometidos a varias pruebas para poder calificar qué tan predecibles son éstos [3].

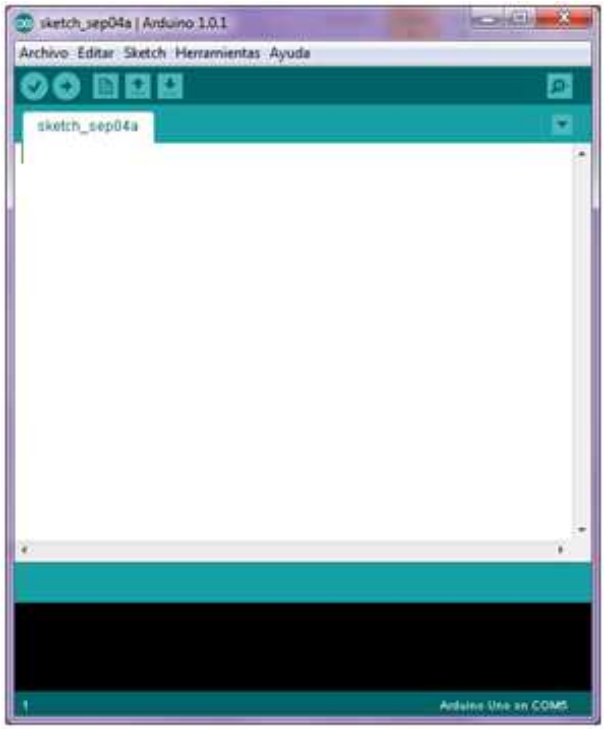


Figura 2: IDE de Arduino.

Definiciones

El ruido es una señal analógica formada por una mezcla aleatoria de longitudes de onda, no contiene información y puede ser producido por el ambiente (temperatura, acústica, perturbación, etc.). La hipótesis planteada en este trabajo consiste en generar la semilla del generador a través de una variación eléctrica proveniente del exterior y que el microcontrolador adquirirá para iniciar la secuencia de números.

Arduino es una plataforma abierta de hardware (figura 1) y software (figura 2) para realizar prototipos de forma rápida y que cuenta con su propio ambiente de desarrollo. *Arduino Uno* es una placa o tarjeta electrónica basada en el microcontrolador *ATmega328*. Cuenta con 14 pines digitales de entrada/salida, de las cuales 6 se puede utilizar como salidas *PWM*, 6 entradas analógicas, un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, un conector para ICSP (*In Circuit Serial Programming*), y un botón de *Reset* [4].

Figura 3: Configuración de la placa.

El software también puede ser descargado desde su página oficial [5], existe una versión para Windows, Mac OS x y Linux. También se podrán encontrar enlaces para el manual de instalación y uso del Entorno de Desarrollo Integrado (IDE), así como de la programación que se utiliza (basado en Wiring) y ejemplos (mismos que se incluyen al instalar el IDE).

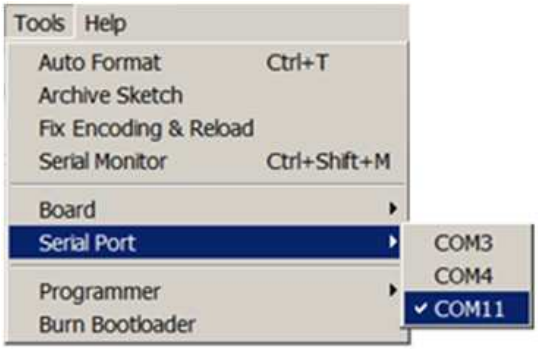


Figura 4: Configuración del puerto serial.

Una vez conectado el Arduino Uno, instalado el driver y el IDE, hay que abrir este último para configurar tanto el tipo de placa (figura 3) que

se va a programar, como también el puerto serial (figura 4).

Desarrollo del generador de números pseudoaleatorios

Etapa 1

Conectamos nuestra tarjeta Arduino Uno a la computadora por medio del cable USB, abrimos nuestra a aplicación y seleccionamos uno de sus pines analógicos como entrada, lo hacemos por medio de nuestro programa, empleando la función pinMode (pin, INPUT), esta entrada físicamente no se conecta a nada, con la función *analogRead ( )* leemos continuamente el valor de este pin, y lo que se va a leer es ruido externo, esto se traduce en nuestra aplicación como un valor entero que va de 0 hasta 1023 (a razón de 1024 valores diferentes). El valor que se lee del pin depende del ruido por lo que no lo podemos determinar, obteniendo así una semilla para nuestro generador de números pseudoaleatorios. La semilla es importante para tener siempre un valor diferente.

Descripción de la función:
analogRead(): Lee el valor de voltaje en el pin analógico especificado. Convertirá voltajes entre 0 y 5 voltios a un número entero entre 0 y 1023.
Sintaxis: ***analogRead(pin)***
Parámetros: pin: Indica el número del pin de la entrada analógica que deseamos leer
Devuelve: int (0 a 1023).

Etapa 2. Generación del número aleatorio:
En nuestro programa utilizamos ahora la función *randomSeed (seed)* tomando como valor de la semilla el que se obtiene de leer el ruido en nuestro pin analógico (etiquetado como A0, refiriéndonos a la Figura 6). A continuación empleamos la función *random (255)* para generar nuestro número aleatorio. El parámetro de la función (255) le indica que los números van en un intervalo de 0 a 255, el cero va implícito como valor mínimo.

La función *random ( )* lee del valor que le pasa *randomSeed (seed)* y genera un número aleatoriamente, ya que se genera el numero se imprime en nuestro monitor.

En esta etapa la lista que se genera de números aleatorios es interminable ya que la función *loop()* donde se encuentra nuestro código se llama indefinidamente.

Descripción de las funciones:
1.- ***randomSeed():*** (seed en inglés, semilla) se utiliza para inicializar el generador de números aleatorios, empezando en un punto arbitrario de su secuencia aleatoria.
Sintaxis: ***randomSeed(seed)***.
Parámetros: **long, int** - recibe un número para generar la semilla.
Devuelve: nada.

2.- ***random():*** La función random genera números pseudoaleatorios.
Sintaxis: **random(max)**, **random(min, max)**.
Parámetros: **min** - límite inferior del valor aleatorio (opcional), **max** - límite superior del valor aleatorio, exclusive (se devuelve hasta el anterior).
Devuelve: un número aleatorio entre min y max (long).

Figura 5: Sketch del Proyecto.

Etapa 3

Se utiliza un push button convencional para acotar una cadena de ocho números aleatorios por serie, cada que se presiona el botón, la cantidad de números que deseamos se controla a través de programación por medios de ciclos y condiciones; se programa el formato de cómo queremos que se muestren (decimal, hexadecimal, binario, en uno o varios renglo-

nes, etc.) en la pantalla de nuestra PC.

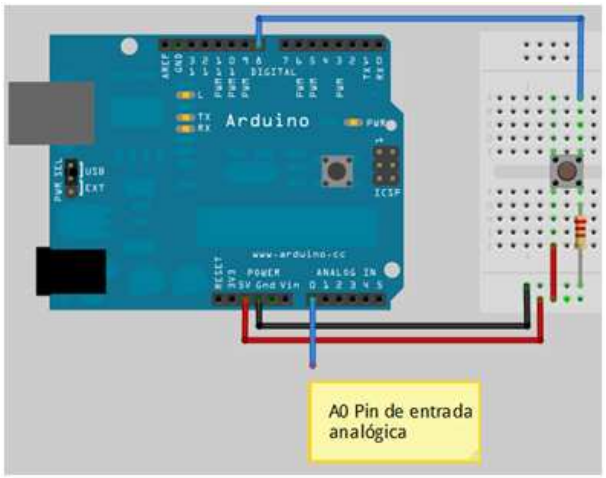


Figura 6: Diagrama de conexiones.

Código del Programa para generar números aleatorios

Un *sketch* es el nombre que se le da a los programas en la plataforma Arduino. La estructura de un sketch es muy similar a la de los programas de lenguaje C, es decir se consta de una función principal, librerías, variables, declaración de funciones etc. Las funciones primordiales en un sketch son *setup ( )* y *loop ( )*. La función *setup( )* es llamada una vez, cuando inicia el sketch. Esta función se ejecutará en una sola ocasión, después de cada reset o encendido del dispositivo. Por su parte, la función *loop( )* es llamada reiteradamente y es el corazón de los sketches, comparable con la función *main ( )* de un código en lenguaje C. En la figura 5 se lista el código diseñado para el sketch generador de números pseudoaleatorios.

La figura 6 muestra el diagrama de conexiones que se realizaron para cumplir con este proyecto. Este diagrama se realizó utilizando el Software de simulación *Fritzing Beta* [6]. Nótese el pin con cable volado (A0), que representa la entrada analógica por donde el microcontrolador adquiere el ruido para generar la semilla aleatoria. Una aproximación al detalle del *pinout* del microcontrolador se aprecia en la figura 7. Finalmente, en la figura 8, se muestra una fotografía del prototipo básico.

Resultados

Una vez que se ejecuta el programa, en la pantalla de la PC es posible revisar los 8 números pseudoaleatorios que generó la tarjeta Arduino Uno. En

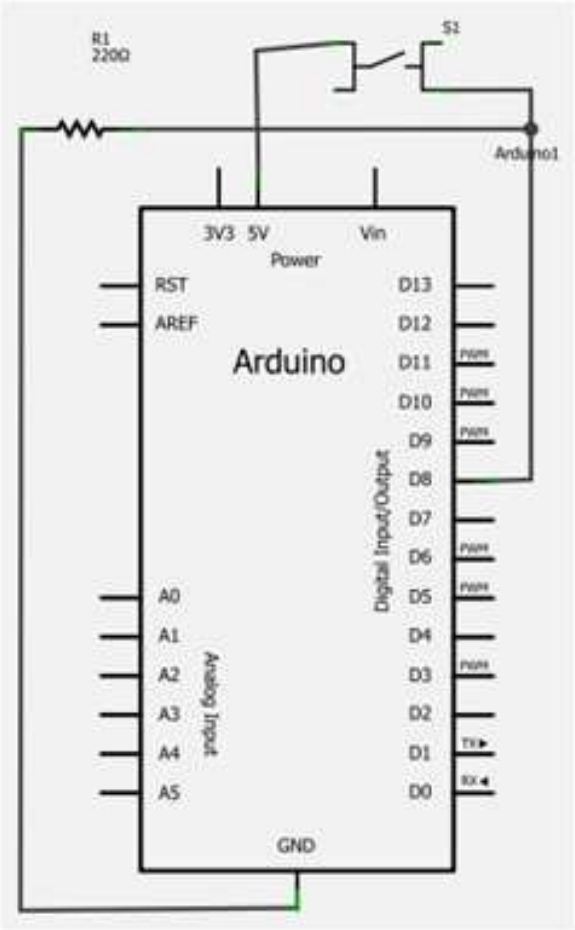


Figura 7: Diagrama esquemático.

la Figura 9 se muestra una impresión de pantalla en donde se listan los números de 8 bits, en base hexadecimal.

El experimento presentado en este trabajo consistió en realizar 10 corridas diferentes (ejecuciones), a las cuales denominamos Series con la finalidad de reportar la Tabla 1.

Conclusiones

En este proyecto se diseñó y probó experimentalmente, a través de una propuesta empírica, un generador de números pseudoaleatorios en un microcontrolador. El generador tiene la particularidad de inicializarse a través de una semilla aleatoria, la cual es producida utilizando ruido eléctrico, garantizando que por lo menos al inicio de la secuencia se tenga una condición no determinística.

Este módulo generador formará parte de un proyec-

Tabla 1. Tabla de resultados.

Serie 1	**Serie 2**	**Serie 3**	**Serie 4**	**Serie 5**	**Serie 6**	**Serie 7**	**Serie 8**	**Serie 9**	**Serie 10**
2D	A	C	F	FE	5A	29	33	40	74
44	67	80	90	64	11	61	46	65	30
5B	C9	B3	67	85	96	78	20	79	21
72	EB	73	6D	60	71	97	48	99	33
A0	82	40	53	74	67	55	D	92	37
B7	78	24	16	19	77	34	47	39	64
CE	57	50	18	58	67	25	5	88	83
FC	66	91	13	60	83	71	36	74	17

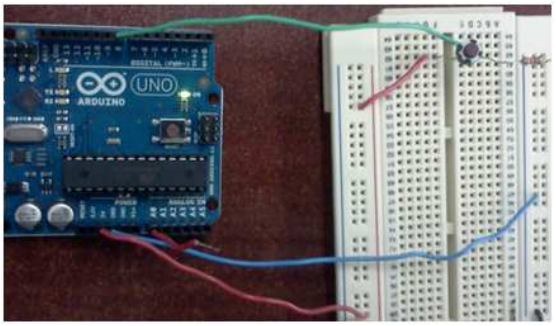


Figura 8: Vista física del Proyecto.

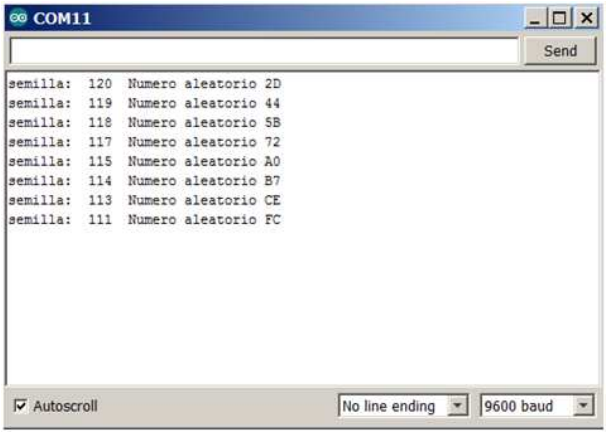


Figura 9: Pantalla de Resultados.

to mayor en donde se programarán algoritmos evolutivos con ejecución intrínseca en un microcontrolador, estos algoritmos son poblacionales y cada individuo de la población se genera aleatoriamente; para una mayor referencia se recomienda consultar [7]. Con respecto al sistema embebido con base en un microcontrolador, se seleccionó la tarjeta Arduino Uno, ya que es sencillo familiarizarse con su entorno de desarrollo, y los recursos disponibles para prototipos rápidos son muy adecuados.

Agradecimientos

Los autores agradecen el apoyo recibido de la Secretaría de Investigación y Posgrado del Instituto Politécnico Nacional (IPN - SIP) a través del proyecto con registro 20131307.

Referencias

1. Tim Riley, *Beautiful Testing: Leading Professionals Reveal How They Improve Software (Theory in Practice)*, O'Reilly Media, ISBN-13: 978-0596159818, 2009.
2. J. C. Herrera-Lozada, Sistema inmune artificial con población reducida para optimización numérica, tesis doctoral, IPN - Centro de Investigación en Computación, 2011.
3. James McNalley, *A Small Book of Random Numbers: 1, CreateSpace Independent Publishing Platform*, ISBN-13: 978-1452818368, 2010.
4. Hoja de datos de la tarjeta de desarrollo Arduino Uno. Enlace consultado por última vez el 29 de abril de 2013. http://arduino.cc/en/Main/ArduinoBoardUno
5. Software de la tarjeta de desarrollo Arduino Uno. Enlace consultado por última vez el 29 de abril de 2013. http://arduino.cc/en/Main/Software
6. Software de simulación para la tarjeta de desarrollo Arduino Uno. Enlace consultado por última vez el 29 de abril de 2013. http://fritzing.org/download/
7. Una introducción a los algoritmos bioinspirados y su implementación en Hardware. *Journal Boliviano de Ciencias*, Vol.8, No.25, 2012. Enlace consultado por última vez el 29 de abril de 2013. http://investigacion.univalle.edu/index.php/publicaciones/revistas/journal-boliviano-de-ciencias/203-journal-boliviano-de-ciencias-25